**Rapport projet : Caraterisation des capteurs**

**Mars/Avril 2019**

Junior TAGNE, Lahat DEME, Louis KABINDA, Loïc TANG

Une image contenant intérieur

Description générée automatiquement



Table des matières

[**I/ Introduction** 3](#_Toc7740481)

[**II/ Objectifs du projet** 3](#_Toc7740482)

[**III/ Réalisation du banc d’essai** 4](#_Toc7740483)

[**IV/ Recherches sur les capteurs** 5](#_Toc7740484)

[**V/ Software de test** 8](#_Toc7740485)

[a) Prérequis 9](#_Toc7740486)

[b) Utiliser l’application 9](#_Toc7740487)

[**VI/ Test et interprétations** 13](#_Toc7740488)

[**VII/ Organisation** 23](#_Toc7740489)

[**VIII/ Conclusion** 23](#_Toc7740490)

## **I/ Introduction**

Dans le cadre de notre 2ème année de cycle ingénieur à l’ISEN nous devons réaliser un projet durant 6 semaines à la fin de notre semestre de manière à mettre en application nos connaissances, les enseignements acquis durant l’année et également continuer à approfondir nos capacités à travailler en groupe qui sont primordiaux pour notre futur métier d’ingénieur.

Notre Groupe est composé de Junior TAGNE, Lahat DEME, Louis KABINDA et Loïc TANG et nous avons travaillé durant ces 6 semaines sur le projet capteurs.

## **II/ Objectifs du projet**

Dans le cadre d’un projet européen, nous devons faire la caractérisation de deux capteurs de proximité : un capteur infrarouge et un capteur ultrasonore et à terme, intégrer ces deux capteurs en parallèle pour une meilleure efficacité sur un fauteuil roulant motorisé et avertir l’utilisateur avec un signal de manière à le rassurer lors de son utilisation.

Notre objectif est donc de caractériser ces capteurs et étudier le comportement des capteurs en fonction des différentes situations que peut rencontrer une personne au quotidien. Pour cela nous avons choisi d’étudier le comportement des deux capteurs selon différentes situations :

* En fonction de la température : effectivement l’utilisateur peut en fonction des saisons ou de la région ou il habite se retrouver face à des climats et donc des températures qui varient on étudiera donc l’influence de la température sur la précision des mesures effectués sur les deux capteurs.
* En fonction de l’humidité : Comme pour la température c’est une variable du climat comme il peut être variable par exemple en fonction des régions ou bien encore des jours de pluie nous avons décidé d’étudier ce facteur.
* En fonction de la luminosité ambiante : l’utilisateur sera amené à utiliser son fauteuil dans des environnement ou la luminosité est variable. La différence de luminosité entre le jour et la nuit ou encore la différence de luminosité entre l’extérieur et l’intérieur d’où l’importance d’étudier le comportement des capteurs dans ces situations.
* En fonction de la surface pointé : Dans l’environnement de son utilisateur le capteur sera confronté à tout type de surfaces et de couleurs différentes, par conséquent on a choisi d’étudier l’influence des matériaux et également des couleurs dans le but d’avoir des informations sur leurs réactions.

## **III/ Réalisation du banc d’essai**

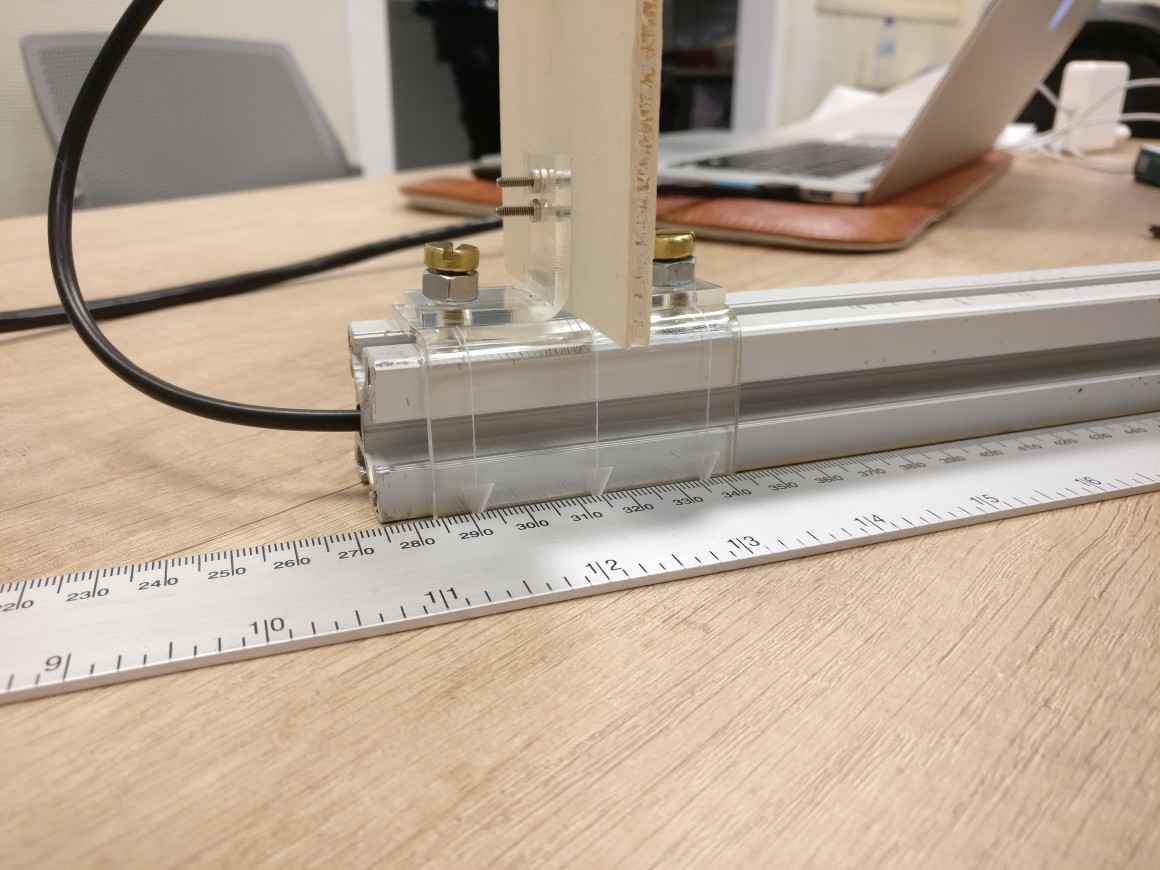
Pour réaliser nos tests nous avons dû élaborer un plan qui nous paraissait le plus adéquat pour réaliser nos tests. Nous un banc d’essai à la manière d’un banc d’optique avec les capteurs qui se trouve à l’extrémité et un cavalier mobile qui soutient une planche de bois avec un angle droit ou seront fixé les surfaces. Le cavalier se déplace sur un profilé et un système d’écrou au niveau du cavalier permet de bloquer la position de celui-ci sur le profilé. Une flèche gravée sur le plexiglass du cavalier nous indique la distance entre le capteur et la surface grâce l’utilisation d’un réglet.



Cavalier

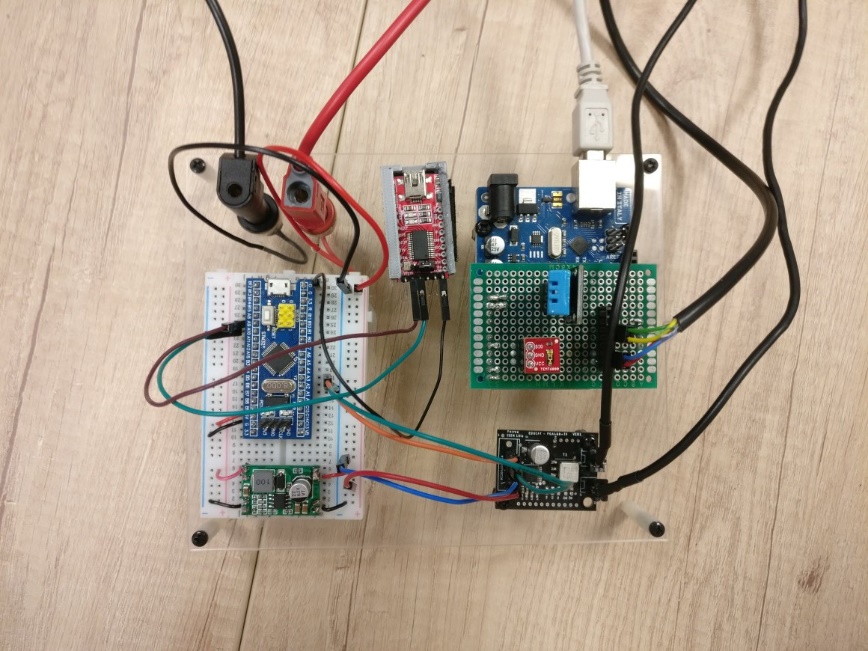
Support capteurs

Photo du banc d’essai de 30 cm



Ecrou de serrage pour le cavalier Flèche indicatrice

Pour l’acquisition des données liés à l’environnement nous avons choisi d’utiliser Arduino et des capteurs complémentaires dont nous avions besoin. Le choix d’Arduino nous est paru comme une évidence car grâce aux librairies disponibles nous avions pu récupérer les informations que les capteurs envoyaient de manière simple. Pour les capteurs, nous avons pris un capteur DHT-11 qui nous permet d’avoir les données de température et d’humidité sur un même capteur, un capteur de luminosité TEMT6000 et un capteur RGB TCS34725 pour la couleur de la surface. Les capteurs de température et de luminosité sont sur le socle comprenant toutes les cartes électroniques (ci-dessous) et le capteurs RGB se trouve sur la planche soutenue par le cavalier.



Socle comprenant l’Arduino et les capteurs et l’électronique des capteurs infrarouges et ultrasonores

Le socle à été réalisé par Ali OUKRID et comprend un emplacement pour la breadboard, pour les systèmes de filtrages et d’amplification pour les capteurs, un emplacement pour l’Arduino et un sdernier pour l’alimentation externe des cartes. Le tout est directement branché sur les capteurs qui se trouvent sur le cavalier et l’extrémité du banc d’essai.

## **IV/ Recherches sur les capteurs**

1. Les capteurs

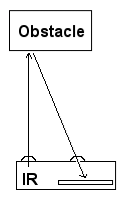
  Les capteurs sont des composants de la chaîne d'acquisition dans une chaîne fonctionnelle.

Les capteurs prélèvent une information sur le comportement de la partie opérative et la transforment en une information exploitable par la partie commande.

Une information est une grandeur abstraite qui précise un événement particulier parmi un ensemble d'événements possibles. Pour pouvoir être traitée, cette information sera portée par un support physique (énergie)on parlera alors de signal. Les signaux sont généralement de nature électrique ou pneumatique.

1. Capteur Infrarouge VL53L1X :

Le capteur SHARP fonctionne en mesurant l'angle de réflexion d'une émission d'IR modulée, grâce à une rangée de récepteur.



* Étendu de mesure :

Le VL53L1X dispose de trois modes de distance (DM): court, moyen et long.

Le mode longue distance permet d’atteindre la distance la plus longue possible de 4 m. Cependant, cette distance maximale de télémétrie est influencée par **la lumière ambiante**.

Le mode courte distance est plus immunisé contre la lumière ambiante, mais sa distance maximale de télémétrie est généralement limitée à 1,3 m.

**Maximum distance vs. Distance mode under ambient light**

|  |  |  |
| --- | --- | --- |
| **Distance mode** | **Max. distance in dark (cm)** | **Max. distance under strong ambient light (cm)** |
| Short | 136 | 135 |
| Medium | 290 | 76 |
| Long | 360 | 73 |

Conditions de test: timing budget = 100 ms, white target 88 %, dark = no IR ambient, ambient light = 200 kcps/SPAD. (Kcps = Khz).

* La rapidité :

Le temps de réponse est de l’ordre des millisecondes.

* La directivité :

La directivité́ est bien meilleure (cône de 5°). Pour faire mieux, il faut ensuite passer à des télémètres laser beaucoup plus chers !

* La consommation :

Seulement 25 mA

* La précision :

La précision du capteur dépend de la distance. Excellente à 10 cm, elle régresse de plus en plus jusqu'à 80cm.

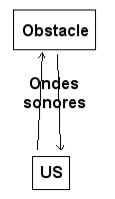
Chaque capteur VL53L1X présente une précision de 1 mm avec une précision de +/- 5 mm et une distance de lecture minimale de ce capteur de 4 cm. Le champ de vision pour cette petite évasion est assez étroit à 15 °, -27 ° avec un taux de lecture allant jusqu'à 50Hz.

* Sensibilité aux interférences et aux autres capteurs :

Ces capteurs IR ont une modulation qui les affranchit normalement de l'éclairage ambiant.

1. Capteur ultrasons 14C01 :

Les capteurs ultrasons fonctionnent en mesurant le temps de retour d'une onde sonore inaudible émise par le capteur. La vitesse du son dans l'air étant à peu près stable, on en déduit la distance de l'obstacle.



* Etendue de mesure :

Quelques mètres en général pour les systèmes ultrasons, même si en théorie il n'y a pas de limite. Il y a aussi en général une distance minimale.

* La rapidité :

Le temps de réponse reste aussi de l’ordre des millimètres <= 1.2 ms.

* La directivité :

Les ultrasons sont très évasifs. Ce qui peut être un gros avantage (détection d'obstacle rapprochée sur une large couronne) ou un gros inconvénient (détection des murs d'un couloir et non du fond du couloir).

* La consommation :

100 mA en veille et jusqu'à plusieurs Ampères en émission.

* La précision :

La précision des ultrasons dépend de la mesure précise du temps de parcours de l'onde sonore. Ce dernier peut aussi varier suivant les conditions de température, de pression ...

* Sensibilité aux interférences et aux autres capteurs :

On l'a vu plus haut, les capteurs ultrasons sont sensibles à la température et à la pression. Mais il y a plus grave : Ils sont aussi sensibles aux autres appareils utilisant les mêmes fréquences, comme les téléobjectifs à ultrasons, ou tout simplement les autres capteurs.

* Spécification

|  |  |  |
| --- | --- | --- |
| Center frequency | 40±1.0KHz |  |
| Echo sensitivity | ≥350mV |  |
| Decay time | ≤1.2ms |  |
| Directivity | 115±15° | X-axis |
| 65±10° | Y-axis |
| Capacitance | 2000±15%pF |  |
| Input voltage | ≤140Vp-p | 40KHz, pulse width 0.5ms, interval 20ms |
| Mean time to failure | 50000h |  |
| Operating temperature | -40~+80℃ |  |
| Storage temperature | -40~+85℃ |  |

## **V/ Software de test**

En parallèle de la réalisation du banc nous avons réfléchit à la manière la manière d’implémenter une application pour automatiser la réalisation des tests sur les capteurs. Les flux de données des différents capteurs seront ainsi récupérés et stocker dans des fichiers au format csv. Pour ce faire, nous avons choisi comme langage de programmation d’utilisé le Node JS. Ce choix est basé sur sa capacité à récupérer des flux de données de manière asynchrone et d’avoir ainsi une récupération de données en temps réel. La première version de l’application est faite en mode console et pourra par la suite être implémenté en mode graphique. Dans ce présent rapport, nous allons expliquer comment utiliser l’application en mode console.

### Prérequis

L’application console étant implémenté en Node JS, il faut tout d’abord:

* Installer Node JS sur son site : <https://nodejs.org/en/>. Attention si vous utilisez une distribution linux les dépôts ne donne une version récente de Node JS et l’application ne pourra pas fonctionner. Pour installer la dernière version, suivre les instructions présentes dans le lien suivant <https://github.com/nodesource/distributions/blob/master/README.md>
* Installer les paquets nécessaires à l’application en faisant un coup de npm install sur la console ce qui a pour objectif d’installer toutes les dépendances du projet

### Utiliser l’application

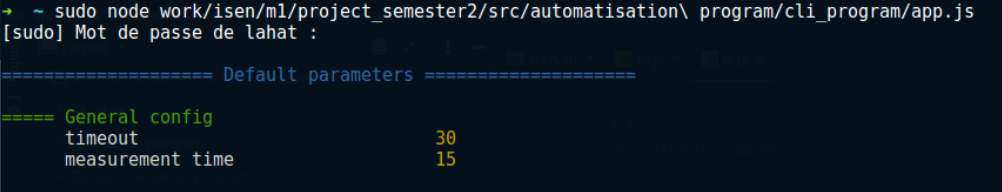
Pour le démarrer, il faut aller dans le répertoire de l’application et utiliser la commande node app.js

**Attention**: Pour certains systèmes d’exploitation (notamment linux), le programme doit être démarré avec les droits d’administrateur parce que ce programme accède aux informations d’un périphérique pour récupérer le flux sortant.

**Remarque:** On peut aussi démarrer l’application en allant dans le dossier « automatisation program/cli\_program/bin » et cliquer sur fichier start-windows-cli.bat pour les utilisateurs Windows et start.sh pour les « Unix like » (Mac OS, Linux, …)

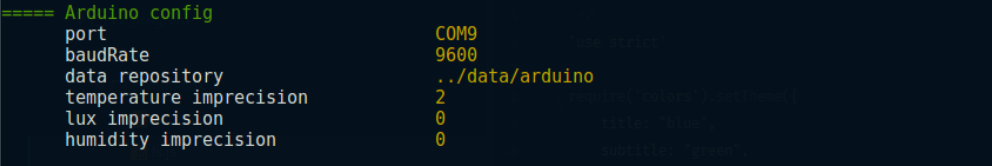
Après avoir démarrer l’application, cette dernière affiche les configurations par défaut qu’il compte utiliser sur les différents capteurs et la liste des environnements par défaut déjà présent dans l’application.

Nous avons tout d’abord les configurations générales de l’application



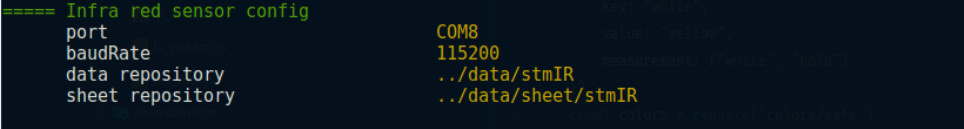
* Le « timeout » désigne le temps que l’application va laisser au testeur pour changer la valeur de la propriété dans l’environnement. Par exemple si pendant le test, on veut tester l’influence de la température sur les capteurs, que le « timeout » est de 120 secondes et que le test se fait sur les environnements suivants : T° = 15, T°30, T°45 alors l’application donnera au testeur 120 secondes pour mettre la température à 15°C pour commencer les tests. Quand l’application aura fini de récupérer les données pour cet environnement, il laissera au testeur encore 120 secondes pour mettre la température à 30°C et commencer les tests. Et c’est la même chose pour la température à 40°C. cependant, si la température atteint la bonne valeur avant que la valeur du « timeout » ne soit atteinte, la récupération de données commence à ce moment.
* Le « measurement time » est le temps de récupération des données. Cela veut dire que pour un environnement donné et un « measurement time » égale à 120 secondes, l’application va enregistrer 120 secondes de données.

Nous avons ensuite la configuration pour le capteur ultrason

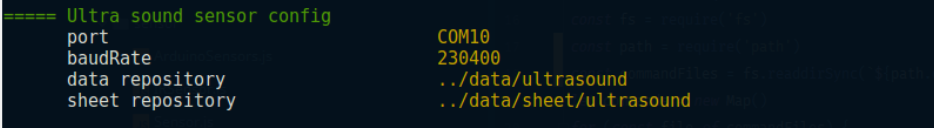


Le « port » désigne le chemin système du port série de l’Arduino. Par exemple /dev/ttyXXX pour MAC/Linux ou COM1 pour Windows.

* Le « baudRate » fait référence à la vitesse de transmission du port série.
* Le « data repository » fait référence au répertoire ou sera stocké les données par rapport au répertoire de l’application
* Le « xxx imprecision » fait référence au capteur sur l’Arduino. Dans notre cas, le capteur de température à une imprécision de plus ou moins de 2°C. Donc lorsque l’application vérifiera si la température actuelle est égale à celle de l’environnement de test , elle cherchera en vérité si elle n’est pas comprise entre T° - 2 et T° + 2.



* Le « port » désigne le chemin système du port série du STM32 qui héberge le capteur infra rouge. Par exemple /dev/ttyXXX pour MAC/Linux ou COM1 pour Windows.
* Le « baudRate » fait référence à la vitesse de transmission du port série.
* Le « data repository » fait référence au répertoire ou cela stocker les données par rapport au répertoire de l’application
* Le « sheet repository » fait référence au répertoire ou sera stocké les courbes des données qui ont été analysé par rapport au répertoire de l’application

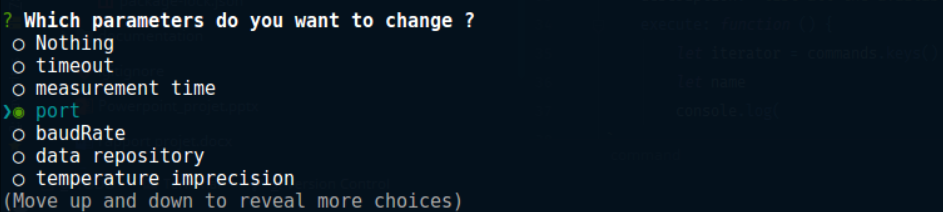


* Le « port » désigne le chemin système du port série de l’ultrason. Par exemple /dev/ttyXXX pour MAC/Linux ou COM1 pour Windows.
* Le « baudRate » fait référence à la vitesse de transmission du port série.
* Le « data repository » fait référence au répertoire ou cela stocker les données par rapport au répertoire de l’application
* Le « sheet repository » fait référence au répertoire ou sera stocké les courbes des données qui ont été analysé par rapport au répertoire de l’application

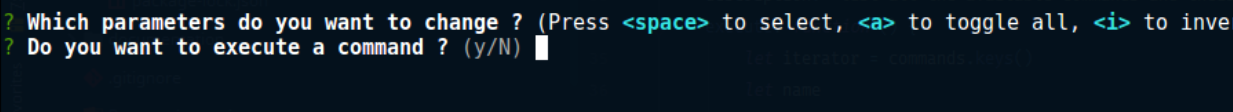
Ensuite l’application affiche les environnements de test en ce moment présent dans l’application ainsi que la propriété qui doit varier sur ces environnements.



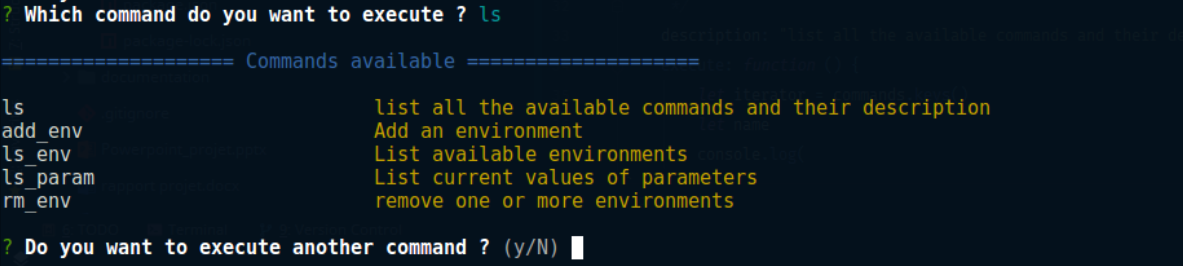
L’application demande ensuite les paramètres à modifier pour ce nouveau test et pour chaque paramètre sélectionné, elle donne la possibilité de changer sa valeur.



Il est ensuite demandé si l’on veut exécuter une commande de l’application ( ls, ls\_env, ls\_param, rm\_env, add\_env)

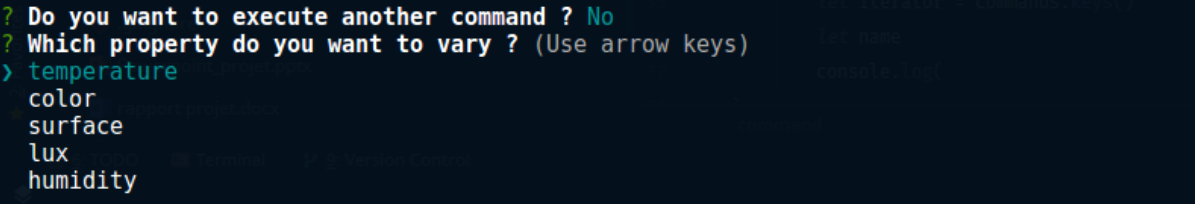




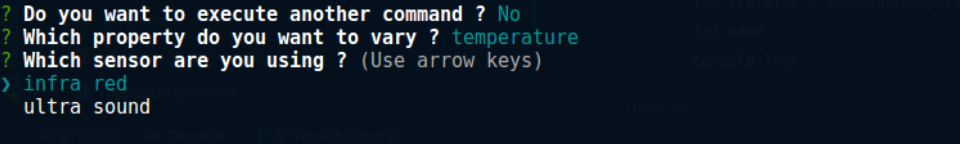


* La commande « ls » permet de lister tous les paramètres et de donner leur définition à côté.
* La commande « ls\_env » permet de lister tous les environnements disponibles en ce moment sur l’application.
* La commande « ls\_param » permet de lister les paramètres de l’application.
* La commande « rm\_env » permet de supprimer un environnement de l’application.
* La commande « add\_env » permet d’ajouter un environnement dans l’application.

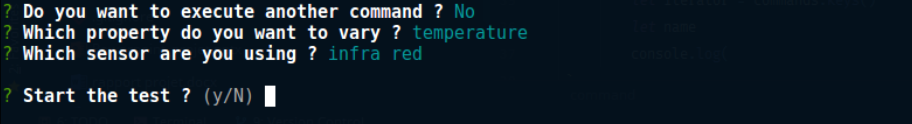
L’application demande ensuite quelle propriété doit changer lors du test pour pouvoir choisir les environnements crées pour le test sur cette propriété.



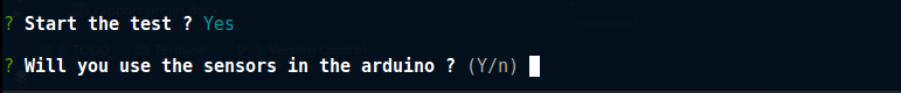
Il est ensuite demandé sur quel capteur voulons-nous faire le test (Infra rouge ou ultrason)



Après ces configurations nous pouvons maintenant commencer les tests.



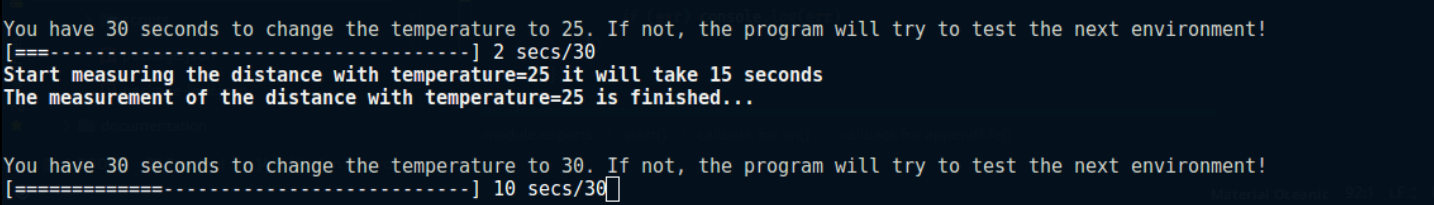
Il sera demandé en dernier si l’on veut utiliser l’Arduino pour automatiser nos tests.



Si oui, les capteurs sur l’Arduino comme le capteur de température, d’humidité et de lumière serviront pour débuter les tests lorsque la valeur de la propriété qui doit varier aura atteint la valeur spécifiée dans l’environnement de test. Et si au bout du temps spécifié par le paramètre « timeout », la propriété n’est pas modifiée correctement, l’application continue le test avec l’environnement suivant.

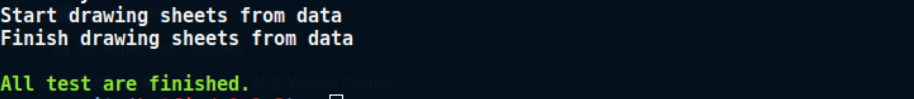
Sinon l’application attendra toujours que le temps spécifié dans le paramètre « timeout » soit atteint avant de commencer les tests et cela même si la valeur la propriété n’est pas correct.

**Remarque**: l’automatisation n’a pas pu être effectué pour la propriété couleur car les données renvoyées par l’Arduino sont au format RGB et nous n’avons pas réussi à déterminer par exemple qu’une certaine valeur de RGB donne du rouge. Pour l’instant la valeur de la couleur est notée en chaîne de caractère dans les environnements de l’application. Une classe « Color » a été implémentée pour les améliorations futur.



Après avoir fait les tests sur les différents environnements et stocker les données dans des fichiers csv, l’application lance un programme python qui va prendre chaque fichier csv et tracer une courbe avec en abscisse la distance mesurée par le capteur (infra rouge ou ultrason) et en ordonnée le nombre d’occurrences.

Après cela, l’application signale la fin des tests et s’arrête.



## **VI/ Test et interprétations**

Une fois le banc et le programme réalisé, nous pouvions commencer les tests pour les tests de surface nous avons réalisé les tests à température ambiante

Tests infrarouges :

Informations sur le test :

Température ambiante : 27°c

L’obstacle se trouve à une distance de 25 cm

Nombre de mesures totales : 1052

|  |
| --- |
| Couleur verte |
| Couleur bleue |
| Couleur blanche |
| Couleur jaune |
| Couleur marron |
| Couleur noire |
| Couleur orange |
| Couleur rose |
| Couleur violet |

Ce que l’on peut en déduire :

Les couleurs jaune et verte ont des courbes similaires (courbe rouge et bleue) et on donc un moyenne distance très proche.

Les couleurs bleu, noire, rose, violet, ont des courbes qui se superposent impeccablement. Et on donc la même distance moyenne.

On pourrait penser que certaines couleurs n’altèrent pas les mesures du capteur, tandis que d’autre justement ont plus d’impact, de l’ordre de 2 ou 4 centimètres par rapport à la mesure réelle de la distance. Une autre remarque c’est que la gaussienne même si et très large pour le courbe et montre donc une irrégularité du capteur.

Le capteur infrarouge, n’est pas très précis et sa précision dépend en parti de l’environnement.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Courbe** | **Couleur plaquette** | **Max (mm)** | **Min(mm)** | **Moyenne** |
| *Bleue* | Vert | 224 | 217 | 221 |
| *Grise* | Rose | 222 | 214 | 218 |
| *Maron* | Noir | 222 | 213 | 218 |
| *Orange* | Bleu | 222 | 214 | 217 |
| *Rose* | Violet | 219 | 213 | 216,5 |
| *Rouge* | Jaune | 225 | 217 | 221 |
| *Verte claire* | Blanche | 222 | 215 | 219 |
| *Violette* | Marron | 224 | 216 | 220 |

Tests ultrasonores :

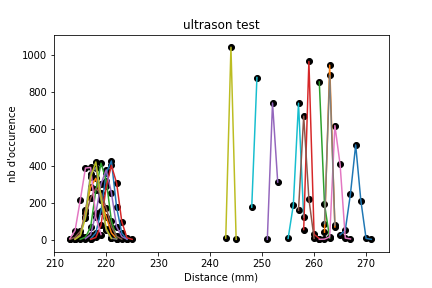
|  |
| --- |
| Courbe mousse verte en bleu |
| Courbe mousse noire en orange |
| Courbe plexiglass en vert |
| Courbe contre-plaqué |
| Une image contenant capture d’écran  Description générée automatiquement  Courbe de la mousse |

On peut déduire que l’a précision est très bonne par rapport au capteur infrarouge, car les pics sont plus marqués en nombre d’occurrences et les courbes sont moins large. De plus, le capteur à une précision au mini mettre près.

Après nous avons eu des résultats surprenants. En effet, pour le même matériel (mousse), on a des piques et des moyennes différentes alors que la couleur ne devrait pas influencer la mesure de la distance. Pour le plexiglass, la mesure de la distance et inférieur à celle qui devait être mesuré par rapport aux autres courbes et à leur moyenne.

Pour la mousse, le capteur trouve une distance infinie, c’est bien à ce résultat que l’on s’attendait. Cela montre bien les matériaux qui composeront les obstacles on beaucoup d’influence sur les capteurs ultrasons.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Courbe** | **Différents matériaux** | **Min** | **Max** | **Moyenne** |
| *Bleue* | Mousse verte | 255 | 258 | 256,5 |
| *Orange* | Mousse noire | 251 | 253 | 252 |
| *Verte* | Plexiglass | 243 | 245 | 244 |
| *Rouge* | Contre-plaqué | 248 | 249 | 248,8 |
|  | Mousse acoustique | … | … | Infini |



Cependant on peut aussi dire :

Les objets que nous avons utilisés sont tous plats, nous n’avons pas pris en compte l’influence de l’angle sur le capteur alors que nous savons qu’elle n’est pas négligeable.

On peut parler sur le fait que pour les courbes du capteur ultrason, pour le même matériau et des couleurs différentes, nous avons eu des distances mesurées différentes pour les deux, alors que les deux mesures auraient dû être identiques. On peut donc penser que nous avons eu des imprécisions avec le banc d’essai.

Nous avons aussi une erreur constante avec entre les distances mesurées entre les infrarouges et les ultrasons : environ 3 centimètres.

Le dernière point reste que nous avons fait les mesures sur des petites distances et qu’il aurait peut-être intéressant de voir les limites que les capteurs nous imposaient.

## **VII/ Organisation**

Pour l’organisation du projet nous avons utilisé Trello c’est une plateforme qui nous permettait de savoir les tâches que chacun avait réalisé, les tâches qui sont en cours de réalisations et les taches qui sont à réaliser. On déterminait ce qu’il y avait à faire en organisant des petites réunions chaque semaine et nous les notions sur Trello.



Au niveau de la répartition des tâches Junior et Loïc se sont occupés de la conception du banc d’essai et de la documentation. Lahat s’est occupé principalement du software d’automatisation des tests et Louis s’est occupé du programme de qui se charge de tracer automatiquement les courbes et de l’analyse de celles-ci.

Les codes sources et autre documentation concernant le projet ont été mis sur GitHub pour y avoir accès au besoin. Le lien vers le projet est <https://github.com/vongolalahad/Projet-M1-S2.git> .



## **VIII/ Conclusion**

Malgré les problèmes que l’on a rencontrés par rapport à la compréhension du sujet sur les premières semaines du projet. Nous avons réussi à remplir nos objectifs qui étaient de caractériser les capteurs infrarouges et ultrasonores, pour cela nous avons réalisé un software automatisant les tests et un banc d’essai. Nous avons réalisé certains tests dans certains environnements et d’autres tests pourront être effectué afin d’avoir plus de résultats et avoir une caractérisation plus précise sur les capteurs. Le projet nous a également permis de nous améliorer dans les différents langages de programmations (python, javascript, node.js).